canvas + JavaScript (utilizzo di ARRAY di Oggetti) **= ANIMAZIONE worm**

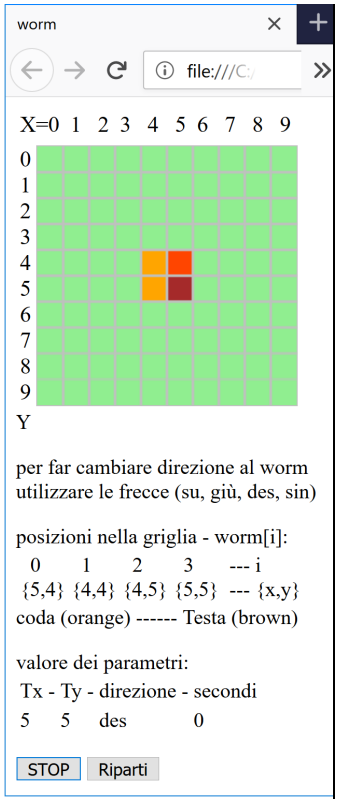
(versione semplificata del gioco *snake* su: https://stackoverflow.com/questions/45588516/javascript-snake-game-apple-respawn)

Il questo esempio si è utilizzato il ***canvas*** per disegnare una scacchiera 10x10 sulla quale far muovere un ***worm*** composto da 4 quadrati (la testa è colorata in **marrone**, la coda in **arancioscuro**); il *worm* si sposta 1 volta al secondo, inizialmente verso il bordo destro del *canvas*; per cambiare verso vanno utilizzati i **tasti di direzione della tastiera →**

codice HTML: ***worm.html***

```
<html>
 <head><title>worm</title></head>
 <body>
   <table><tr><td><span style="font-size:17px">
      X=0  1   2 
      3   4   5  6  
      7   8   9</span>
   </td></tr></table>
   <table><tr><td><span style="font-size:17px">
      0 <br>1 <br>2 <br>3 <br>4 <br>5
      <br>6 <br>7 <br>8 <br>9</span>
   </td>
    <td>
      <canvas id="tela"  width="200"  height="200"
      style="border:1px solid SILVER;"></canvas>
    </td>
   </tr></table>
   <div>Y
      <p>per far cambiare direzione al worm<br>
      utilizzare le frecce (su, gi&ugrave;, des, sin)</p>
      <div>posizioni nella griglia - worm[i]:</div>
      <div id="wr">
          <table><tr>
            <td>  0<br>{0,0}</td>
            <td>  1<br>{0,0}</td>
            <td>  2<br>{0,0}</td>
            <td>  3<br>{0,0}</td>
            <td> --- i<br> --- {x,y}</td>
          </tr></table>
      </div>
      <div>coda (orange) ------ Testa (brown)</div>
      <p>valore dei parametri:
      <table><tr>
           <td>Tx</td>  <td> - </td>
           <td>Ty</td>  <td> - </td>
           <td>direzione</td> <td> - </td>
           <td>secondi</td>
        </tr><tr>
           <td id="Tx">x</td>   <td></td>
           <td id="Ty">y</td>   <td></td>
           <td id="VS">dir</td> <td></td>
           <td id="sec">0</td>
      </tr></table>
      </p>
      <button onclick="stop()" >STOP</button>
      <button onclick="start()">Riparti</button>
   </div>
   <script type="text/JavaScript"  src ="worm.js"> </script>
 </body>
</html>
```

Il *tag* ***<canvas>*** **è contenuto** nella seconda **cella di una tabella** HTML formata da 1 sola riga e 2 colonne (la prima colonna contiene i valori **0**,**1**,…**9** che indicano lo spostamento verticale (**Y**) del *worm* nella griglia (*all'inizio la coda è 4 riquadri in basso rispetto al* ***bordo alto*** *del canvas indicato con* **Y=0**); il *tag* <*canvas*> è preceduto da **un'altra tabella** formata da 1 sola riga e 1 cella che contiene i valori **0**,**1**,…**9** che indicano lo spostamento orizzontale (**X**) del *worm* nella griglia (*all'inizio la coda è 5 riquadri a destra rispetto al* ***bordo sinistro*** *del canvas indicato con* **X=0**)



Nel codice HTML dopo il *canvas* ci sono le **tabelle** per esporre il valore delle variabili JS ad ogni secondo:

- posizioni nella griglia dei 4 quadrati di ***worm*** realizzato in JS come un ARRAY di Oggetti **{x,y}** worm[**0**] rappresenta la **coda**, worm[**3**] la **Testa**;

- **valore dei parametri** utilizzati nel codice JavaScript: **Tx**, **Ty**, **direzione**, **secondi** trascorsi dall'inizio della animazione (valore che indica anche il numero degli spostamenti di ***worm*** dalla posizione iniziale).

Seguono i 2 pulsanti **STOP** e **Riparti** (*tag* ***<button>***) per bloccare l'animazione e riattivarla successivamente dal punto in cui si era fermata (per ripartire dall'inizio occorre usare il tasto Aggiorna del browser).

Segue il ***tag <script>*** per il **codice JS** che in questo esempio **NON è scritto direttamente nel <*body*>** del file HTML ma **in un file separato** (nell'esempio il codice è scritto nel file **worm.js**).

codice JS: ***worm.js*** (1^ parte)

```
   var N= 10   //---numero di quadrati di riga e colonna
   var Q= 20;  //---dimensione in pixel dei quadrati

   var Tx= Ty= 5;//---riquadro occupato inizialm.da Testa

//--- spostamento di riquadro rispetto ai bordi del canvas:
//--- orizzontale (variazione di x), verticale (variazione di y)
   var xVar= yVar= 0;
   var dir;     //---direzione scelta con freccia (su, giù, des, sin)

   var worm = [];  //---ARRAY worm------

//--- riattivazione game ogni 1000 millisecondi (1 volta al secondo)
   var tempo = 1000,      secs  = 0;

   canv= document.getElementById("tela");
   ctx=canv.getContext("2d");

   document.addEventListener("keydown",cntrTasto);

//---inserimento nell'ARRAY delle posizioni iniziali:
      worm.push( {x: Tx,   y: Ty-1} );  //..0 (di coda)
      worm.push( {x: Tx-1, y: Ty-1});  //..1
      worm.push( {x: Tx-1, y: Ty}  );   //..2
      worm.push( {x: Tx,   y: Ty}  );   //..3 (di Testa)

//--- all'inizio simula scelta key=39 ... freccia a destra (+X)
      xVar= 1;
      dir = "des";

      timer = setInterval(game,tempo);

function cntrTasto (evt) {  //--- controllo tasto premuto
      switch(evt.keyCode) {
        case 37:             //--- left arrow
             xVar= -1;       //--- a sinistra (-X)
             yVar= 0;
             dir = "sin";
               break;
        case 38:             //--- up arrow
             xVar= 0;
             yVar= -1;       //--- in alto   (-Y)
             dir = "su";
               break;
        case 39:             //--- right arrow
             xVar= 1;        //--- a destra  (+X)
             yVar= 0;
             dir = "des";
               break;
        case 40:             //--- down arrow
             xVar= 0;
             yVar= 1;        //--- in basso  (+Y)
             dir = "gi&ugrave;";
               break;
      }   //--- fine istruzione switch ... case
}         //--- fine della funzione cntrTasto() ---
function stop() {
      clearInterval(timer);
}
function start() {
      timer = setInterval(game,tempo);
}
```

DESCRIZIONE del codice JS

DEFINIZIONE delle **variabili globali**
(riconosciute da tutte le funzioni scritte nel file *.js*):

Tx rappresenta, ad ogni secondo, lo scostamento della **Testa** di *worm* dal ***bordo sinistro*** del *canvas* (spostamento orizzontale) e **Ty** dal ***bordo in alto*** (spostamento verticale); **xVar** e **yVar** indicano se la **Testa** dovrà spostarsi di un riquadro:

- a destra (**xVar= 1**) o a sinistra (**xVar= -1**)
- in basso (**yVar= 1**) o in alto (**yVar= -1**)

L'array **worm** viene riempito (mediante il metodo ***.push*** degli Array) con **4 Oggetti** che hanno solo 2 proprietà: **{x,y}** entrambe numeriche; ogni Oggetto rappresenta il riquadro occupato nella griglia dal quadrato di ***worm*** (**x** lo scostamento del quadrato dal ***bordo sinistro*** del *canvas*, **y** lo scostamento dal ***bordo in alto***). *NOTA: i 4 Oggetti hanno solo le proprietà* **x** *e* **y** *e sono* ***privi di metodi*** *propri.*

L'array è riempito a partire dalla **coda** (elemento **0**) l'ultimo elemento inserito è la **Testa** (posizione **3**).

Il **penultimo quadrato** di **worm** (a posizione **2**, prima di **Testa**) occupa, inizialmente, il riquadro alla stessa **quota** (scostamento dal ***bordo alto***) di **Testa** (**y: Ty**) ma più vicino al ***bordo sinistro*** (**x:Tx-1**, cioè **x** vale 4); la **coda** invece è alla stessa **ascissa** di **Testa** (**x: Tx**) ma ad **altezza** inferiore (**y:Ty-1**, cioè **y** vale 4).

Il metodo ***addEventListener(…)*** dell'oggetto ***document*** permette di impostare la **funzione** da richiamare (in questo caso **cntrTasto**) al verificarsi dell'**evento** specificato: in questo esempio viene gestito l'evento **keydown** (intercettato dal *browser* quando si **preme un tasto** – **key** – della tastiera).
Elenco degli eventi e descrizioni sono disponibili su:
https://www.w3schools.com/jsref/dom_obj_event.asp

Premuto un tasto, il *browser* attiverà la funzione **cntrTasto(…)** passandole un oggetto (che in questo esempio è stato chiamato **evt**) che ha una proprietà, **keyCode**, che contiene il valore (nella codifica *Unicode*) del carattere corrispondente al tasto premuto sulla tastiera. Il programma gestisce solo i valori **37**, **38**, **39** e **40** corrispondenti ai **tasti di direzione della tastiera**.

L'istruzione **switch … case** si usa quando occorre scegliere una strada fra numerose ***alternative*** in base a diversi valori di una variabile (in questo esempio, in base al valore della proprietà **keyCode** dell'oggetto **evt**). L'istruzione **break** chiude il rispettivo **case** (e non vanno indicate le parentesi graffe); l'istruzione è simile alla sequenza di **if… else if…** , potrebbe scriversi:

```
if (evt.keyCode == 37) {
   xVar= -1;   yVar= 0;  dir = "sin";
} else
if (evt.keyCode == 38) { .... } else if ....     e così via.
```

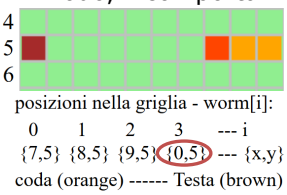
codice JS: *worm.js* (2^ parte)

```
function game() {
   disegna_campo();
   aggiornaParmVideo();
//--- incremento coordinate di Testa per la prossima attivazione
   Tx += xVar;    //--- equivalente a: Tx = Tx + xVar;
   Ty += yVar;    //--- equivalente a: Ty = Ty + yVar;

//---controllo che Testa stia all'interno del canvas----
   if (Tx < 0)  {          //-- se fuori dal bordo sinistro
      Tx =  N-1;           //-- Testa spunterà da destra
   }
   if (Tx > N-1)  {        //-- se fuori dal bordo a destra
      Tx = 0;              //-- Testa spunterà da da sin
   }
   if (Ty < 0)  {          //-- se fuori dal bordo in alto
      Ty = N-1;            //-- Testa spunterà dal basso
   }
   if (Ty >  N-1)  {       //-- se fuori dal bordo in basso
      Ty = 0;              //-- Testa spunterà in alto
   }
//---disegno di worm sul canvas ------------------------
   for(var i=0; i<worm.length; i++) {
    if ( i == worm.length - 1)     //--colore di Testa
            ctx.fillStyle="BROWN";
    else
    if ( i == 0)          //--- colore di Coda (posizione 0)
            ctx.fillStyle="ORANGERED";
    else                  //--- colore del corpo (posizioni 1 e 2)
            ctx.fillStyle="ORANGE";
    ctx.fillRect(worm[i].x *Q +1 , worm[i].y *Q +1 , Q-2 , Q-2 );
   }

   worm.shift();          //---avanzamento worm---
   worm.push( {x:Tx,   y:Ty} );
   secs ++;
}                         //--- fine della funzione game()---
function disegna_campo(){
   ctx.fillStyle  = "LightGREEN";  //--- sfondo del campo
   ctx.fillRect (0,0,  canv.width,  canv.height);

   ctx.lineWidth     = "1";     //--- spessore e colore linee

   ctx.strokeStyle  = "SILVER";

                              //--- 9 colonne da canv.height pixel
   for(var col=1; col<N; col++){ //--- 9 colonne
         ctx.moveTo(col*Q,  0);        //---  a 10, 20..80px
         ctx.lineTo  (col*Q,  canv.height);   //---da X=0
         ctx.stroke();
   }                          //--- 9 righe da canv.width pixel
   for(var rig=1; rig<N; rig++){
         ctx.moveTo(  0       , rig*Q);  // a 10, 20..80px
         ctx.lineTo(canv.width, rig*Q);  //da bordo alto Y
         ctx.stroke();
   }
}
function aggiorna_HTML (elemento, valore){
   oggetto=  document.getElementById(elemento);
   oggetto.innerHTML  =  valore;
}
```

DESCRIZIONE del codice JS

La **funzione** **game()**, attivata con il metodo ***setInterval(…)*** dell'oggetto ***window***, **effettua:**

1. il **disegno** del campo e della griglia,
2. l'aggiornamento dei parametri esposti a video nella pagina web,
3. agg.to dei parametri **Tx** e **Ty** (aumentandoli o riducendoli di 1 in base al valore di **xVar** e **yVar**)
4. la verifica che le coordinate di **Testa** (cioè i parametri **Tx** e **Ty**) alla prossima attivazione restino nella griglia del *canvas* (nell'intervallo da **0** a **9**, cioè da **0** a **N-1**); in caso contrario vengono portate ai valori dell'estremo opposto dell'intervallo (simulando, in questo modo, il comportamento di una superficie sferica:



raggiunto il bordo a destra (posizione **{9,5}**) vengono impostate per **Testa** le coordinate **{0,5}** così **Testa** spunterà da sinistra!

5. il **disegno** sul ***canvas*** con il metodo **.fillRect** dei 4 quadrati di **worm** (impostando il vertice in alto a sin del quadrato **i** a: **worm[i].x** * Q + 1 **pixel** dal bordo a sin. e a **worm[i].y***Q+1 **px** dal bordo in alto; larghezza e altezza del rettangolo pari a **Q−2** (**pixel**, cioè 18px, in modo da non sovrapporlo ai quadrati da 20px della griglia)
6. l'**avanzamento** di **worm** realizzato con:

- lo **slittamento all'indietro** delle coordinate dei 4 segmenti con il metodo ***.shift()*** degli array che elimina da **worm** l'oggetto contenuto nella posizione 0 (worm[**0**]) e fa arretrare di una posizione gli altri segmenti: [1] va in [0], [2] in [1] , [3] –cioè la precedente posizione di Testa- va in [2],
- l'**inserimento** con metodo ***.push()*** in posizione 3 delle **nuove coordinate** di **Testa** (**Tx** e **Ty**)

codice JS: *worm.js* (3^ parte)

```
function aggiornaParmVideo(){
   aggiorna_HTML ("sec", secs);
   aggiorna_HTML ("Tx", Tx);
   aggiorna_HTML ("Ty", Ty);
   aggiorna_HTML ("VS", dir);
//---costruzione per concatenazione (operatore +)
//--- del testo che rigenera la table nella div id="wr"
 var tab =  "<table><tr>";
  for (var i=0; i<worm.length; i++) {
   tab +="<td>  "+ i +"<br>{";
   tab += worm[i].x + ",";
   tab += worm[i].y + "}</td>";
  }
  tab +="<td> --- i"
  tab +="<br> --- {x,y}</td>";
  tab +=  "</tr></table>";
  aggiorna_HTML ("wr", tab);
}
```